Proyecto Final

Algoritmo - Fabian Vinicio Constante Nicolalde

Joshua Alomoto – Bryan Cando

Introducción

Para escribir el código proporcionado, varios conocimientos en el ámbito de la programación en el lenguaje C y la manipulación básica de entrada/salida. Aquí hay una descripción de los conocimientos utilizados:

Lenguaje C: El código está escrito en C, por lo que fue necesario tener un conocimiento sólido de la sintaxis, estructuras de control de flujo (como bucles y condicionales), operaciones aritméticas y declaraciones de variables.

Entrada y salida en C: Utilicé funciones como printf y scanf para interactuar con el usuario, mostrando mensajes en la consola y leyendo la entrada del usuario.

Operaciones aritméticas: Calculé el cambio y la cantidad de monedas utilizando operadores aritméticos básicos, como la resta y la división.

Estructuras de control de flujo: Implementé bucles (do-while) para validar las entradas del usuario y asegurarme de que no ingresaran valores negativos.

Uso de variables: Declaré y utilicé variables para almacenar y manipular datos, como el total a cobrar, el monto que paga el cliente, el cambio y la cantidad de monedas.

Condicionales: Utilicé una estructura if para verificar si el cliente ha pagado lo suficiente y proporcionar mensajes de error en caso contrario.

Bucles: Implementé bucles while para calcular la cantidad de monedas necesarias para el cambio.

Formato de salida: Utilicé printf para formatear y mostrar la salida de manera clara y comprensible.

Manejo de errores: Incluí verificaciones para asegurarme de que los valores ingresados no fueran negativos, y proporcioné mensajes de error y solicitudes de entrada adicional en esos casos.

Ejercicio Principal

En pocos meses se instalará un sistema electrónico para el pago de estacionamiento de la nueva plaza comercial de la ciudad. Se colocarán máquinas donde el usuario introduce su boleto y el dispositivo que lo recibe envía el dato a un programa que se encarga de calcular el cambio. Se requiere elaborar un programa que, por lo pronto, pida ambos datos del teclado: el total a cobrar por el estacionamiento y monto que paga el cliente (este valor debe llegar automáticamente puede ser por mensaje sms al usuario). El programa sólo puede dar monedas de 10, 5, 2 y 1 peso y se requiere que el dispositivo entregue el menor número de monedas posible para alargar el tiempo requerido entre una carga de dinero y la siguiente. Deberá indicar el cambio que va a dar, cuántas monedas de cada denominación y el total de monedas que entregará.

Resolución Algorítmica En Pseint

```
Algoritmo PagoEstacionamiento
  Definir totalCobrar, montoCliente, cambio, monedas10, monedas5, monedas2, monedas1 Como Real
  Definir tipoMoneda Como Cadena

  // Pedir al usuario el total a cobrar y verificar que no sea negativo
  Repetir
    Escribir "Introduzca el total a cobrar por el estacionamiento en pesos: ";
    Leer totalCobrar;
    Si totalCobrar < 0 Entonces
      Escribir "El total a cobrar no puede ser negativo. Intente nuevamente.";
    FinSi
  Hasta Que totalCobrar >= 0;

  Repetir
    Escribir "Introduzca el monto del cliente en pesos: ";
    Leer montoCliente;
    Si montoCliente < 0 Entonces
      Escribir "El monto del cliente no puede ser negativo. Intente nuevamente.";
    FinSi
  Hasta Que montoCliente >= 0;

  Escribir "El monto del cliente se ha registrado automáticamente.";
```

```
  cambio = montoCliente - totalCobrar;

  // Verificar si el cambio es en pesos o dólares
  Si cambio < 0 Entonces
    Escribir "Monto insuficiente. El cliente debe pagar al menos ", totalCobrar, " pesos.";
  Sino
    Escribir "Cambio a entregar en pesos: ", cambio, " pesos";
    Escribir "Monedas de 10 pesos: ", monedas10;
    Escribir "Monedas de 5 pesos: ", monedas5;
    Escribir "Monedas de 2 pesos: ", monedas2;
    Escribir "Monedas de 1 peso: ", monedas1;
    Escribir "Total de monedas en pesos: ", monedas10 + monedas5 + monedas2 + monedas1;

    // Convertir el cambio a dólares (suponiendo una tasa de cambio fija)
    cambio = cambio / 20.0; // Suponiendo que 1 dólar = 20 pesos
    tipoMoneda = "dólares";

    Escribir "Cambio a entregar en", tipoMoneda, ": ", cambio, " ", tipoMoneda;
  FinSi
FinAlgoritmo
```

Declaración de variables

Se comienza declarando las variables necesarias para el programa. En este caso, se definen variables como totalCobrar, montoCliente, cambio, monedas10, monedas5, monedas2, y monedas1 como variables de tipo real

Solicitud del total a cobrar

Se utiliza un bucle Repetir...Hasta Que para solicitar al usuario el total a cobrar por el estacionamiento. Se verifica que el total no sea negativo, y se muestra un mensaje de error en caso contrario.

Solicitud del monto del cliente

Se utiliza otro bucle Repetir...Hasta Que para solicitar al usuario el monto que paga el cliente. Se verifica que el monto no sea negativo, y se muestra un mensaje de error en caso contrario.

Registro automático del monto del cliente

Se simula que el monto del cliente se ha registrado automáticamente por mensaje SMS.

Cálculo del cambio

Se calcula el cambio restando el total a cobrar del monto del cliente.

Verificación del pago suficiente

Se verifica si el cambio es negativo, lo cual indicaría que el cliente no ha pagado lo suficiente. En este caso, se muestra un mensaje indicando que el monto es insuficiente.

Cálculo de las monedas

Si el cambio es positivo, se inicializan variables para contar las monedas de diferentes denominaciones (10, 5, 2, 1 pesos). Se utiliza un bucle Mientras para calcular la cantidad de monedas necesarias.

Mostrar el cambio y la cantidad de monedas

Finalmente, se muestra el cambio a entregar y la cantidad de monedas de cada denominación, así como el total de monedas.

Resolución Algorítmica En Lenguaje C

```c
#include <stdio.h>

int main() {
    double totalCobrar, montoCliente;

    // Pedir al usuario el total a cobrar y verificar que no sea negativo
    do {
        printf("Ingrese el total a cobrar en dólares: ");
        scanf("%lf", &totalCobrar);

        if (totalCobrar < 0) {
            printf("El total a cobrar no puede ser negativo. Intente nuevamente.\n");
        }
    } while (totalCobrar < 0);

    // Pedir al usuario el monto que paga el cliente y verificar que no sea negativo
    do {
        printf("Ingrese el monto que paga el cliente en dólares: ");
```

```
    scanf("%lf", &montoCliente);

    if (montoCliente < 0) {
        printf("El monto que paga el cliente no puede ser negativo. Intente nuevamente.\n");
    }
} while (montoCliente < 0);

// Calcular el cambio
double cambio = montoCliente - totalCobrar;

// Verificar si el cliente ha pagado lo suficiente
if (cambio < 0) {
    printf("Monto insuficiente. El cliente debe pagar al menos %.2f dólares.\n", totalCobrar);
    return 1; // Salir del programa con error
}

// Inicializar variables para contar las monedas
int monedas10 = 0, monedas5 = 0, monedas2 = 0, monedas1 = 0;

// Calcular el cambio con las monedas disponibles
while (cambio >= 10) {
    cambio -= 10;
    monedas10++;
}

while (cambio >= 5) {
    cambio -= 5;
    monedas5++;
}

while (cambio >= 2) {
    cambio -= 2;
    monedas2++;
}

while (cambio >= 1) {
    cambio -= 1;
    monedas1++;
}

// Mostrar el cambio y la cantidad de monedas
printf("\nCambio a entregar: %.2f dólares\n", montoCliente - totalCobrar);
printf("Monedas de 10 dólares: %d\n", monedas10);
printf("Monedas de 5 dólares: %d\n", monedas5);
printf("Monedas de 2 dólares: %d\n", monedas2);
printf("Monedas de 1 dólar: %d\n", monedas1);
printf("Total de monedas: %d\n", monedas10 + monedas5 + monedas2 + monedas1);

return 0; // Salir del programa exitosamente
```

```
}
}
```

Declaración de variables

Se declaran las variables necesarias para almacenar el total a cobrar, el monto que paga el cliente, el cambio y las cantidades de monedas de diferentes denominaciones.

```
float totalCobrar, montoCliente;
    int monedas10 = 0, monedas5 = 0, monedas2 = 0, monedas1 = 0;
```

Entrada de datos

Se utiliza printf y scanf para solicitar al usuario que ingrese el total a cobrar y el monto que paga el cliente.

```
printf("Ingrese el total a cobrar: ");
    scanf("%f", &totalCobrar);
    printf("Ingrese el monto que paga el cliente: ");
    scanf("%f", &montoCliente);
```

Cálculo del cambio

Se calcula la diferencia entre el monto pagado por el cliente y el total a cobrar para obtener el cambio.

```
float cambio = montoCliente - totalCobrar;
```

Manejo de casos insuficientes

Se verifica si el cliente ha pagado lo suficiente y, en caso contrario, se imprime un mensaje de error y se sale del programa.

```
if (cambio < 0) {
        printf("Monto insuficiente. El cliente debe pagar al menos
%.2f pesos.\n", totalCobrar);
        return 1; // Salir del programa con error
    }
```

Cálculo de monedas

Se utiliza un bucle while para calcular la cantidad de monedas de 10, 5, 2 y 1 peso que se deben entregar como cambio.

```
while (cambio >= 10) {
        cambio -= 10;
        monedas10++;
    }
    // Se repite el proceso para las otras denominaciones (5, 2 y 1 peso)
```

Mostrar resultado

Finalmente, se imprime el cambio y la cantidad de monedas de cada denominación, así como el total de monedas.

```
printf("\nCambio a entregar: %.2f pesos\n", montoCliente - totalCobrar);
    printf("Monedas de 10 pesos: %d\n", monedas10);
    // Se repite para las otras denominaciones
    printf("Total de monedas: %d\n", monedas10 + monedas5 + monedas2 + monedas1);
```

Puntos Positivos De Nuestos Codigos

El código proporcionado es una implementación funcional del problema y sigue un enfoque claro y comprensible. Sin embargo, la "mejor" o "más óptima" manera de implementar un programa puede depender de diversos factores, como la legibilidad del código, la eficiencia en tiempo de ejecución, la gestión de errores, entre otros. En términos de eficiencia, el algoritmo utilizado es simple y directo. Se utiliza un enfoque de "greedy" (voraz) para seleccionar la mayor cantidad posible de monedas de mayor valor antes de pasar a denominaciones más pequeñas, lo cual es adecuado para este problema. La optimización del código puede incluir mejoras en la gestión de errores, manejo de casos límite, o incluso la implementación de funciones para facilitar la lectura y mantenimiento del código. Además, la validación de la entrada del usuario podría mejorarse para garantizar que se ingresen valores válidos.